Making a CK file

October 2022

Summary

- **SPICE provides means to create CK files, either by packaging orientation computed by others, or by first computing orientation within SPICE and then packaging it in a CK file**

  - **Packaging of <u>already existing</u> orientation data can be done in two ways:**

    » **Use SPICE CK writer routines by calling them from within your own SPICE-based application**

    » **Convert a text file containing orientation data to a CK using the Toolkit's *msopck* program**

  - **<u>Computing as well as packaging</u> orientation can be done in two ways:**

    » **Use SPICE geometry routines and CK writer routines by calling them from within your own SPICE-based application**

      - **Constructing orientation using SPICE routines is not discussed here**

    » **Convert orientation rules and schedules to a CK using the *prediCkt* program available from the NAIF website**

CK Writer Routines

- **The SPICE toolkit provides the following CK writer routines for the FORTRAN, C, IDL and MATLAB toolkits, respectively:**
  - **For Type 1 CK**
    - » **CKW01 / ckw01_c / cspice_ckw01**
  - **For Type 2 CK**
    - » **CKW02 / ckw02_c / cspice_ckw02**
  - **For Type 3 CK**
    - » **CKW03 / ckw03_c / cspice_ckw03**
  - **For Type 4 CK**
    - » **CKW04B, CKW04A, CKW04E (no CSPICE, Icy, or Mice wrappers)**
  - **For Type 5 CK**
    - » **CKW05 / ckw05_c (no Icy or Mice wrapper)**
  - **For Type 6 CK**
    - » **CKW06 (no CSPICE, Icy or Mice wrappers)**

- **Only the Type 3 writer is discussed in this tutorial**
  - **Writers for Types 1 and 2 have very similar interfaces**
  - **Types 4, 5 and 6 are are not commonly used**

- **The following C-language code fragment illustrates the creation of a Type 3 C-kernel having a single segment.**

```
ckopn_c ( filename, "my-ckernel", 0, &handle );
/*
   Insert code that properly constructs the
   sclkdp, quats, avvs, and starts arrays.
*/
ckw03_c ( handle, begtim, endtim, inst,
          "ref", avflag, "segid", nrec,
          sclkdp, quats, avvs, nints, starts );

ckcls_c ( handle );
```

- **handle** - file handle for the newly created C-kernel.

- **begtim**, **endtim** - start and stop times in SCLK ticks for the segment.

- **inst** - ID code for the instrument for which the C-kernel is being made.

- **ref** - name of the base reference frame. Must be one known to SPICE during your program execution.

- **avflag** - a `SpiceBoolean` indicating whether or not to include angular velocity in the segment.

- **segid** - a string identifying the segment. It must be no more than 40 characters in length.
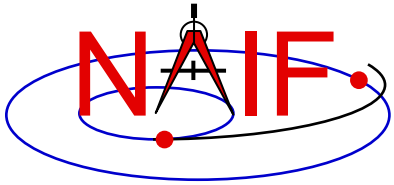
- **`nrec`** - number of records in `sclkdp`, `quats`, and `avvs`.

- **`sclkdp`** - monotonically increasing list of times, given in SCLK ticks, that identify when `quats` and `avvs` were sampled or calculated.

- **`quats`** - a list of SPICE quaternions that rotate vectors from the base frame specified by the `ref` argument to the `inst` frame.

- **`avvs`** - angular rate vectors given in the base frame specified by the `ref` argument.

- **`nints`** - number of entries in starts.

- **`starts`** - a list of SCLK ticks indicating the start of interpolation intervals. They must correspond to entries in `sclkdp`.
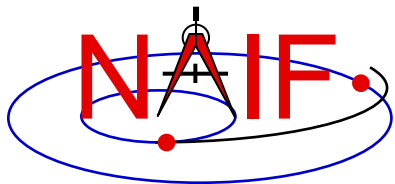
- **One of the easiest ways to make up angular rates is to assume a constant rotation rate between subsequent quaternions:**

```
for(k=0; k<(nrec-1); k++ ) {
  q2m_c ( quats[k][0], init_rot );
  q2m_c ( quats[k+1][0], final_rot );
  mtxm_c ( final_rot, init_rot, rotmat );
  raxisa_c ( rotmat, axis, &angle );
  sct2e_c ( scid, sclkdp[k], &init_et );
  sct2e_c ( scid, sclkdp[k+1], &final_et );
  vscl_c ( angle/(final_et-init_et), axis,
          &avvs[k][0] );                      }
```

- **Then copy the (`nrec-1`) value of `avvs` into the last element of `avvs`.**
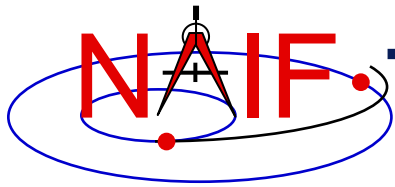
- **Constructing angular rates in this fashion assumes that no more than one 180-degree rotation has occurred between adjacent quaternions.**
  - **`raxisa_c` chooses the smallest angle that performs the rotation encapsulated in the input matrix.**

- **Other techniques exist, including differentiating quaternions. Care must be exercised when taking that approach.**

MSOPCK

- *msopck* makes CK files from orientation data provided in a time- tagged, space-delimited table in text format

- *msopck* can process quaternions (SPICE and non-SPICE styles), Euler angles, or rotation matrices, tagged with UTC, SCLK, or ET time tags

- *msopck* requires all program directives to be provided in a setup file that follows the SPICE text kernel syntax

- *msopck* has a simple command line interface with the following usage

```
msopck setup_file input_data_file output_ck_file
```

- **If the specified output CK already exists, new segment(s) are appended to it**
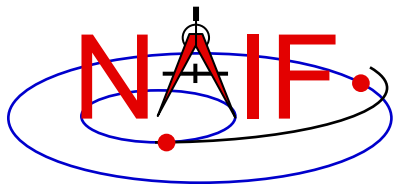
MSOPCK
List of Setup File Keywords
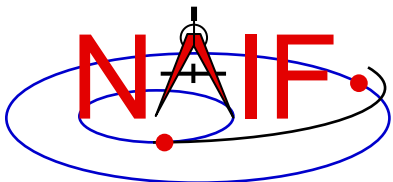
Navigation and Ancillary Information Facility

Supporting

Kernels/Files

```
LSK_FILE_NAME          = 'LSK file'
SCLK_FILE_NAME         = 'SCLK file' (or MAKE_FAKE_SCLK='new SCLK file')
FRAMES_FILE_NAME       = 'FRAMES file'
COMMENTS_FILE_NAME     = 'file containing comments'
```

Output CK

Specifications

```
PRODUCER_ID            = 'producer group/person name'
INTERNAL_FILE_NAME     = 'internal file name string'
CK_SEGMENT_ID          = 'segment ID string'
CK_TYPE                = 1, 2, or 3
INSTRUMENT_ID          = CK ID
REFERENCE_FRAME_NAME   = 'reference frame name'
MAXIMUM_VALID_INTERVAL = interval length, seconds
```

Input data

Specifications

```
INPUT_TIME_TYPE        = 'SCLK', 'UTC', 'TICKS', 'DPSCLK', or 'ET'
TIME_CORRECTION        = bias to be applied to input times, seconds
INPUT_DATA_TYPE        = 'MSOP QUATERNIONS', 'SPICE QUATERNIONS',
                         'EULER ANGLES', or 'MATRICES'
QUATERNION_NORM_ERROR  = maximum normalization error
EULER_ANGLE_UNITS      = 'DEGREES' or 'RADIANS'
EULER_ROTATIONS_ORDER  = ( 'axis3', 'axis2', 'axis1' )
EULER_ROTATIONS_TYPE   = 'BODY' or 'SPACE'
ANGULAR_RATE_PRESENT   = 'YES', 'NO', 'MAKE UP', 'MAKE UP/NO AVERAGING'
ANGULAR_RATE_FRAME     = 'REFERENCE' or 'INSTRUMENT'
ANGULAR_RATE_THRESHOLD = ( max X rate, max Y rate, max Z rate )
OFFSET_ROTATION_ANGLES = (  angle3,  angle2,  angle1 )
OFFSET_ROTATION_AXES   = ( 'axis3', 'axis2', 'axis1' )
OFFSET_ROTATION_UNITS  = 'DEGREES' or 'RADIANS'
DOWN_SAMPLE_TOLERANCE  = down sampling tolerance, radians
INCLUDE_INTERVAL_TABLE = 'YES' or 'NO' (default 'YES')
CHECK_TIME_ORDER       = 'YES' or 'NO' (default 'NO')
```

Optional and conditional keywords are shown in green

Making a CK File

Navigation and Ancillary Information Facility

Four Examples

INPUT_DATA_TYPE = 'SPICE QUATERNIONS'

Input file:
```
TIME1 [TIME2] QCOS QSIN1 QSIN2 QSIN3 [ARX ARY ARZ ]
..... ....... .... ..... ..... .....  ... ... ...
TIME1 [TIME2] QCOS QSIN1 QSIN2 QSIN3 [ARX ARY ARZ ]
```

INPUT_DATA_TYPE = 'MSOP QUATERNIONS'

Input file:
```
TIME1 [TIME2] -QSIN1 -QSIN2 -QSIN3 QCOS [ARX ARY ARZ ]
..... ....... .....  .....  ..... ....  ... ... ...
TIME1 [TIME2] -QSIN1 -QSIN2 -QSIN3 QCOS [ARX ARY ARZ ]
```

INPUT_DATA_TYPE = 'EULER ANGLES'

Input file:
```
TIME1 [TIME2] ANG3 ANG2 ANG1 [ARX ARY ARZ ]
.....  ...... .... .... ....  ... ... ...
TIME1 [TIME2] ANG3 ANG2 ANG1 [ARX ARY ARZ ]
```

INPUT_DATA_TYPE = 'ROTATION MATRICES'

Input file:
```
TIME1 [TIME2] M11 M12 M13 M21 ... M33 [ARX ARY ARZ ]
.....  ...... ... ... ... ... ... ...  ... ... ...
TIME1 [TIME2] M11 M12 M13 M21 ... M33 [ARX ARY ARZ ]
```

Navigation and Ancillary Information Facility

- **Quaternions**
  - **INPUT_DATA_TYPE = 'SPICE QUATERNIONS' indicates the quaternions being used follow the SPICE formation rules(*)**
  - **INPUT_DATA_TYPE = 'MSOP QUATERNIONS' indicates the quaternions being used follow the traditional AACS formation rules(*)**
    - » **Normally quaternions that come in telemetry are of this type**
  - **QUATERNION_NORM_ERROR keyword may be used to identify and filter out input records with quaternions that are not unit vectors**
    - » **It is set to a tolerance for comparing the norm of the input quaternion with 1**

- **Euler angles**
  - **All three angles must be provided**
  - **For the angles provided on the input as**
    ```
    TIME1 [TIME2] ANG3 ANG2 ANG1 [ ARX ARY ARZ ]
    ```
    and rotation axes specified in the setup as
    ```
    EULER_ROTATIONS_ORDER  = ( 'axis3', 'axis2', 'axis1' )
    ```
    the matrix rotating vectors from base to the structure frame is computed as
    ```
    Vinst = [ANG3]axis3 * [ANG2]axis2 * [ANG1]axis1 * Vref
    ```
  - **Angles can be provided in degrees or radians**

    (*) NAIF prepared a "white paper" explaining differences between various quaternion styles:

Navigation and Ancillary Information Facility

- **Angular rates are an optional input. Their presence or absence must be indicated using the ANGULAR_RATE_PRESENT keyword**
  - **If angular rates are provided (ANGULAR_RATE_PRESENT = 'YES'), they must be in the form of a 3D vector expressed either in the base frame (less common) or instrument frame (more common)**
    - » **The ANGULAR_RATE_FRAME keyword must be set to indicate which of the two is used**
  - **If angular rates are not provided, the program can either make a CK without rates (ANGULAR_RATE_PRESENT = 'NO'), or try to compute rates from the orientation data by using a uniform rotation algorithm implemented in Type 3 CKs, either with averaging (ANGULAR_RATE_PRESENT = 'MAKE UP') or without averaging (ANGULAR_RATE_PRESENT = 'MAKE UP/NO AVERAGING') of the rates computed for adjacent orientation data points**
  - **ANGULAR_RATE_THRESHOLD may be used to identify and filter out input records with angular rate components that are too large to be real**

- **Input data can be tagged with UTC, SCLK string, SCLK ticks or ET, as specified using the INPUT_TIME_TYPE keyword**
  - **Time tags must not have embedded spaces**

Navigation and Ancillary Information Facility

- *msopck* can generate Type 1, 2, or 3 CKs
  - Type 1 is rarely used - only in cases when the input contains very few data points that are far apart so that interpolation between them makes no sense
  - Type 2 is also rarely used, primarily to package orientation for spinning spacecraft
    - » Normally the input for making Type 2 CKs should contain two times and the angular rate in each record
  - Type 3 is the most commonly used type because it provides interpolation between the orientation data points stored in the CK

- Interpolation intervals are determined based on the threshold value specified in the MAXIMUM_VALID_INTERVAL keyword
  - The threshold interval is specified in units of seconds
  - A Type 3 CK will allow interpolation between all input points for which the duration between points is less than or equal to the threshold

- An additional transformation to be combined with the input attitude may be specified using OFFSET_ROTATION_* keywords
  - The convention for specification of the offset rotation angles is the same as for the input Euler angles
 - A vector defined in the base frame is first multiplied by the offset rotation

```
Vinst = [ROTinput] * [ROToffset] * Vref
```

MSOPCK - Output Details (2)

- **The time tags may be adjusted by a constant value, specified in seconds, using the TIME_CORRECTION keyword**

- **The order of input time tags can be checked using the CHECK_TIME_ORDER keyword.**

- **The output CK file contains one or more CK segments**
  - **Multiple segments are generated if the input data volume is large and does not fit into the program's internal buffer (100,000 pointing records)**
  - **When the output file has many segments, each segment's start time is equal to the stop time of the previous segment, i.e. there are no gaps at the segment boundaries**

- **The Comment area of the output CK contains the following information:**
  - **Contents of a comment file, if it was specified using the COMMENT_FILE_NAME keyword**
  - **Contents of the setup file**
  - **Summary of coverage for each segment written to the file, including a table listing interpolation intervals for segments of Type 2 or 3**
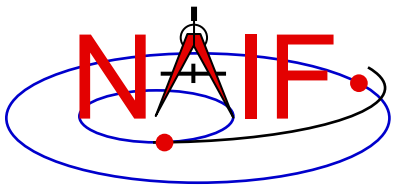
```
Terminal Window

$ more msopck_setup.example
MSOPCK setup for predict M'01 CK generation.
========================================================================
\begindata
   PRODUCER_ID             = 'NAIF/JPL'
   LSK_FILE_NAME           = 'naif0007.tls'
   SCLK_FILE_NAME          = 'ORB1_SCLKSCET.00001.tsc'
   COMMENTS_FILE_NAME      = 'msopck_comments.example'
   INTERNAL_FILE_NAME      = 'sample M01 SC Orientation CK File'
   CK_SEGMENT_ID           = 'SAMPLE M01 SC BUS ATTITUDE'
   INSTRUMENT_ID           = -53000
   REFERENCE_FRAME_NAME    = 'MARSIAU'
   CK_TYPE                 = 3
   MAXIMUM_VALID_INTERVAL  = 60
   INPUT_TIME_TYPE         = 'SCLK'
   INPUT_DATA_TYPE         = 'MSOP QUATERNIONS'
   QUATERNION_NORM_ERROR   = 1.0E-3
   ANGULAR_RATE_PRESENT    = 'MAKE UP'
\begintext
$
```
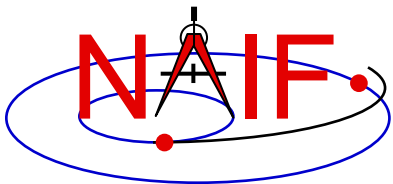
Navigation and Ancillary Information Facility

```
                          Terminal Window                                    ☒

$ more msopck_comments.example


Sample Mars Surveyor '01 Orbiter Spacecraft Orientation CK File
========================================================================


Orientation Data in the File
------------------------------------------------------------


    This file contains sample orientation for the Mars Surveyor '01
    Orbiter (M01) spacecraft frame, 'M01_SPACECRAFT', relative
    to the Mars Mean Equator and IAU vector of J2000, 'MARSIAU', inertial
    frame. The NAIF ID code for the 'M01_SPACECRAFT' frame is -53000.

Status
------------------------------------------------------------


    This file is a special sample C-Kernel file created by NAIF to illustrate
    MSOPCK program. This file should not be used for any other purposes.


...
```
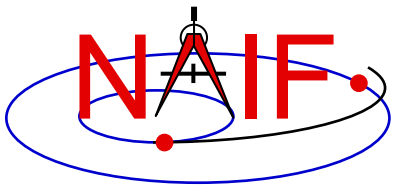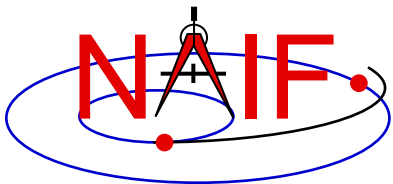
Navigation and Ancillary Information Facility

```
                                                        Terminal Window                                              ☒

$ more msopck_input.example
0767491368.064      -0.24376335      0.68291384      0.28475901      0.62699316
0767491372.114      -0.24249471      0.68338563      0.28591829      0.62644323
0767491373.242      -0.24204185      0.68355329      0.28633291      0.62624605
0767491374.064      -0.24194814      0.68358228      0.28641744      0.62621196
0767491380.064      -0.24012676      0.68424169      0.28807922      0.62543010
0767491386.064      -0.23830473      0.68489895      0.28973563      0.62464193
0767491392.064      -0.23648008      0.68555126      0.29139303      0.62384833
0767491398.064      -0.23465389      0.68620253      0.29304524      0.62304745
0767491404.064      -0.23282999      0.68684150      0.29470173      0.62224580
0767491404.114      -0.23277293      0.68686688      0.29475362      0.62221455
0767491405.242      -0.23231585      0.68702790      0.29516507      0.62201253
0767491410.064      -0.23100059      0.68748174      0.29634561      0.62143935
0767491416.064      -0.22917353      0.68811325      0.29799308      0.62062853
0767491422.064      -0.22734161      0.68874177      0.29963482      0.61981412
0767491428.064      -0.22551078      0.68936246      0.30128030      0.61899473
0767491434.064      -0.22367453      0.68998299      0.30291779      0.61816987
0767491436.114      -0.22300583      0.69021050      0.30351804      0.61786298
0767491438.011      -0.22251770      0.69037871      0.30395477      0.61763631
...
```

```
Terminal Window                                                            ☒

$ msopck msopck_setup.example msopck_input.example msopck_example_ck.bc


MSOPCK Utility Program, Version 3.0.0, 2003-05-05; SPICE Toolkit Ver. N0057

...

<comment file contents>

...

<setup file contents>

...

***********************************************************************

RUN-TIME OBTAINED META INFORMATION:

***********************************************************************

PRODUCT_CREATION_TIME = 2004-04-29T12:17:55

START_TIME            = 2004-04-27T00:00:05.516

STOP_TIME             = 2004-04-27T23:59:56.275

***********************************************************************

INTERPOLATION INTERVALS IN THE FILE SEGMENTS:

***********************************************************************

SEG.SUMMARY: ID -53000, COVERG: 2004-04-27T00:00:05.516 2004-04-27T23:59:56.275

-----------------------------------------------------------------------

     2004-04-27T00:00:05.516    2004-04-27T20:05:26.282

     2004-04-27T20:11:20.278    2004-04-27T23:59:56.273
```

PREDICKT

- *prediCkt* makes CK files from a set of orientation specification rules, and schedules defining when these rules are to be followed

- *prediCkt* has a simple command line interface

- *prediCkt* requires orientation and schedule specifications to be provided in a setup file that follows the SPICE text kernel syntax

- *prediCkt* requires the names of all supporting kernels -- SPK, PCK, etc -- be provided in a meta-kernel (a "furnsh kernel")

- *prediCkt* and its User Guide are available only from the Utilities link of the NAIF webpages
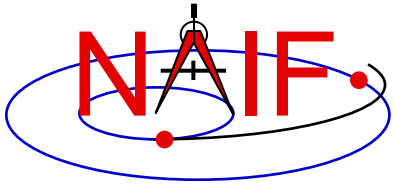
- *prediCkt* has the following command line arguments

      ```
      -furnish support_data

      -spec ck_specs

      -ck outfile

      -tol fit_tolerance [units]

      -<sclk|newsclk> sclk_kernel
      ```

- **'-furnish', '-spec' and '-ck' are used to specify the input meta-kernel, input attitude specification file and output CK file**

- **'-tol' is used to specify the tolerance to which the orientation stored in the CK should match the specified attitude profile**

- **'-sclk' or '-newsclk' specify the name of an existing SCLK or the new "fake" SCLK to be created for use with the output CK**

- **A "FURNSH" kernel lists SPICE kernels that are to be used by prediCkt to determine geometry needed to compute orientations**

- **A prediCkt attitude specification (spec) file, using the text kernel syntax, is used to provide three types of information:**
  - **specification of dynamic directions**
  - **specification of orientations based on these directions**
  - **specification of the schedules defining when those orientations should be followed**

- **The contents of the FURNSH kernel and the spec file are included in the comment area of the output CK file**
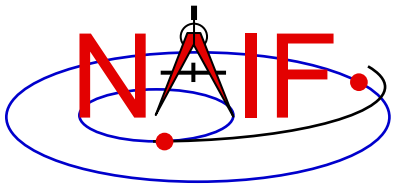
PREDICKT - Directions

- **Dynamic directions can be of the following types**
  - **Based on ephemeris (position vectors, velocity vectors)**
  - **Fixed with respect to a reference frame (expressed as a Cartesian vector or specified by RA and DEC)**
  - **Towards sub-observer point**
  - **Based on the surface normal and lines of constant latitude or longitude**
  - **Based on other, already defined directions (rotated from them, computed as cross products using them, etc)**

- **Example: these two sets of keyword assignments specify nadir and spacecraft velocity directions for the M01 spacecraft**

```
DIRECTION_SPECS      += ( 'ToMars    = POSITION OF MARS -' )
DIRECTION_SPECS      += (              'FROM M01        -' )
DIRECTION_SPECS      += (              'CORRECTION NONE'    )
DIRECTION_SPECS      += ( 'scVelocity = VELOCITY OF M01  -' )
DIRECTION_SPECS      += (              'FROM MARS        -' )
DIRECTION_SPECS      += (              'CORRECTION NONE'    )
```
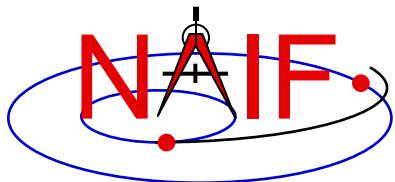
Navigation and Ancillary Information Facility

- ## An orientation is specified by:
  - defining that one of the frame's axes (+X,+Y,+Z,-X,-Y,-Z) points exactly along one of the defined directions
  - defining that another of the frame's axes points as closely as possible to another defined direction
    - » The third axis is the cross product of the first two
  - specifying the base frame with respect to which the orientation of this "constructed" frame is to be computed

- ## Example: these keyword assignments specify the nominal nadir orientation for the THEMIS instrument, flown on the M01 spacecraft

```
ORIENTATION_NAME      += 'CameratoMars'
PRIMARY               += '+Z = ToMars'
SECONDARY             += '+Y = scVelocity'
BASE_FRAME            += 'J2000'
```

- **A schedule is defined by specifying a series of time intervals during which a given orientation is to be followed**

  - **For each interval for a given CK ID, the spec file defines the orientation name, start time, and stop time (as Ephemeris Times)**

- **Example: these spec file keyword assignments specify a schedule with a single window during which M01 (Mars Odyssey) will yield nadir-pointed orientation for the THEMIS instrument**
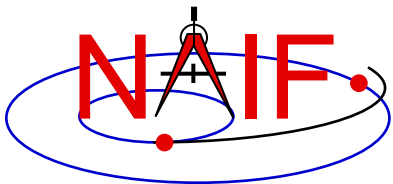
```
CK-SCLK                =  53
CK-SPK                 = -53
CK-FRAMES             += -53000
CK-53000ORIENTATION   += 'SOLUTION TO M01_THEMIS_IR = CameratoMars'
CK-53000START         += @2004-FEB-10-00:00
CK-53000STOP          += @2004-FEB-15-00:00
```

Navigation and Ancillary Information Facility

- **In the example on the previous slide:**

    - **the CK-FRAMES keyword specifies the CK ID to be used in the output CK**
        - » **This ID is incorporated into the keywords defining the schedule intervals**

    - **the CK-SCLK keyword specifies the ID of the SCLK kernel to be used in creating the CK**

    - **the CK-SPK keyword specifies the ID of the object, the position of which is used in applying light time correction when orientation is computed**

    - **the "SOLUTION TO" construct specifies that although the orientation is sought for the M01 spacecraft frame (ID -53000), it is computed for the camera frame (M01_THEMIS_IR) and then transformed to the spacecraft frame**

Navigation and Ancillary Information Facility

```
                                                    Terminal Window                                                    ☒
$ cat m01_map_nadir.prediCkt
\begindata
        DIRECTION_SPECS      += ( 'ToMars    = POSITION OF MARS -' )
        DIRECTION_SPECS      += (                'FROM M01        -' )
        DIRECTION_SPECS      += (                'CORRECTION NONE'  )


        DIRECTION_SPECS      += ( 'scVelocity = VELOCITY OF M01  -' )
        DIRECTION_SPECS      += (                'FROM MARS       -' )
        DIRECTION_SPECS      += (                'CORRECTION NONE'  )


        ORIENTATION_NAME     += 'CameratoMars'
        PRIMARY              += '+Z = ToMars'
        SECONDARY            += '+Y = scVelocity'
        BASE_FRAME           += 'J2000'


        CK-SCLK               =  53
        CK-SPK                = -53
        CK-FRAMES            += -53000
        CK-53000ORIENTATION  += 'SOLUTION TO M01_THEMIS_IR = CameratoMars'
        CK-53000START        += @2004-FEB-10-00:00
        CK-53000STOP         += @2004-FEB-15-00:00
\begintext
```
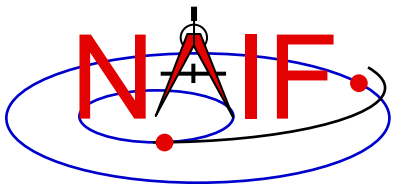
PREDICKT - Example (2)

Navigation and Ancillary Information Facility

```
Terminal Window                                                                    ✕

$ cat m01_map_nadir.furnsh
\begindata
   KERNELS_TO_LOAD = ( 'naif0007.tls'
                       'm01_v26.tf'
                       'mar033-5.bsp'
                       'm01_map_rec.bsp'
                       'm01.tsc' )
\begintext
$ prediCkt -furnish m01_map_nadir.furnsh -spec m01_map_nadir.prediCkt -ck m01_map_nadir.bc -tol
0.01 degrees -sclk m01.tsc


Begin Segment: 1 --- SOLUTION TO M01_THEMIS_IR = CameratoMars


Constructing Segment
From: 2004 FEB 10 00:00:00.000
To  : 2004 FEB 15 00:00:00.000
   Percentage finished:   0.0%
   Percentage finished:   5.0 %    (50 quaternions)
   ...
   Percentage finished:  95.0 %    (925 quaternions)
$
```